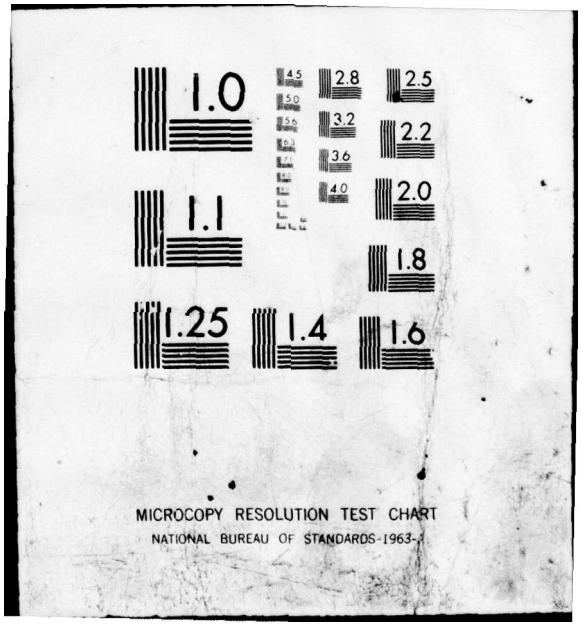
AD-A075 499 NAVAL POSTGRADUATE SCHOOL MONTEREY CA F/G 17/2
AN APPLICATION OF RULE-DIRECTED INTERACTIVE TRANSACTION AGENT (--ETC(U)
JUN 79 M H SMITH

UNCLASSIFIED NL

1 OF 1
AD A075499

END
DATE FILMED
11-79
DDC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-

LEVEL II

(2)

NAVAL POSTGRADUATE SCHOOL

Monterey, California

(12) 86

(9) Master's THESIS

(6) AN APPLICATION OF RULE-DIRECTED INTERACTIVE TRANSACTION AGENT (RITA) FOR THE AUTOMATED TECHNICAL CONTROL OF THE DEFENSE COMMUNICATIONS SYSTEM (DCS)

by

(10) Mark Howard Smith

(11) Jun[redacted]79

Thesis Advisor: G. K. Poock

Approved for public release; distribution unlimited.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) An Application of Rule-directed Interactive Transaction Agent (RITA) for the Automated Technical Control of the Defense Communications System (DCS)		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis; June 1979 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Mark Howard Smith		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		12. REPORT DATE June 1979 13. NUMBER OF PAGES 85
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Rule-directed Interactive Transaction Agent (RITA)
Technical Control Facility (TCF)
Automated Technical Control (ATEC)
Defense Communications System (DCS) ARPANET
System Control (SYSCON)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Historically, military technical control facilities (TCF) have performed their supervisory and control functions over communications-electronics systems by manual processes, using verbal and/or teletype conversations for coordinating with distant stations. These methods will soon be changed with the application of computers to automate many of the manual tasks and assist the technical controllers in others. One of these manual tasks, that

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601
(Page 1)

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

#20 - ABSTRACT - CONTINUED

of record-keeping and report generation, promises to yield significant manpower savings but has been deferred from implementation due to high cost and technical complexity. Accordingly, this thesis proposes a representative software program for an automatic report generator using the Rand Corporation developed production rule system, Rule-directed Interactive Transaction Agent (RITA). It facilitates the automatic entry of data into reporting formats, automatic generation of data into files/reports, a capability to alter report formats and file building rules, and algorithms for automated statistical analysis of data files.

Approved for public release; distribution unlimited.

An Application of Rule-directed Interactive Transaction Agent (RITA) for the Automated Technical Control of the Defense Communications System (DCS)

by

Mark Howard Smith
Major, United States Air Force
B.S., California State University (Long Beach), 1964
M.S., University of Southern California, 1967

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN OPERATIONAL DECISION SYSTEMS

from the

NAVAL POSTGRADUATE SCHOOL

June 1979

Author Mark H Smith

Approved by: Gary Poock
Thesis Advisor

JR Murphy
Second Reader

John M Wozencraft
Chairman, Command, Control, and Communications Academic Group

Jack R Borsting
Academic Dean

ABSTRACT

Historically, military technical control facilities (TCF) have performed their supervisory and control functions over communications-electronics systems by manual processes, using verbal and/or teletype conversations for coordinating with distant stations. These methods will soon be changed with the application of computers to automate many of the manual tasks and assist the technical controllers in others. One of these manual tasks, that of record-keeping and report generation, promises to yield significant manpower savings but has been deferred from implementation due to high cost and technical complexity. Accordingly, this thesis proposes a representative software program for an automatic report generator using the Rand Corporation developed production rule system, Rule-directed Interactive Transaction Agent (RITA). It facilitates the automatic entry of data into reporting formats, automatic generation of data into files/reports, a capability to alter report formats and file building rules, and algorithms for automated statistical analysis of data files.

TABLE OF CONTENTS

I. INTRODUCTION

The author was assigned, for a two year period in the early 1970's, as commander of a major communications facility in the mountains of southern Italy. Mt. Vergine is a primary nodal point in the wideband radio transmission network of the Defense Communications System (DCS) in Europe. Its mission is to provide DCS connectivity to US forces located in the Mediterranean area, including Italy, Greece, and Turkey.

The major communications-electronics facilities at the station are an AUTOVON switching center, several microwave and tropospheric scatter radio equipments, and a primary technical control facility (TCF). Essentially the heart of the station, the TCF function alone required in excess of 30 military personnel.

During this two year period, many preparations were underway (including construction of a new building) in anticipation of a major TCF upgrade program, the World-Wide Technical Control Improvement Program (WWTCIP). Unfortunately, the program never reached fruition due to technical difficulties and contractor default.

A companion TCF improvement effort, however, is still viable and promises significant benefits. The Automated Technical Control (ATEC) program is developing computer-assisted equipments and procedures for the TCF function.

The ATEC benefits of improved customer service, coupled with an accompanying net savings in manpower, certainly has raised the interest of the military services.

The interest of the author has also been raised, especially by the prospects for going even further than the current effort of the ATEC program. This thesis investigates further ATEC refinements, which would yield equally desirable benefits as the current program. The aim of the thesis is to illustrate a methodology for implementing these refinements and to motivate their eventual incorporation into the ATEC program.

II. BACKGROUND

Growing weary of complaints from military communications customers, US Air Force communications planners launched an effort in the late 1960's to diagnose system degradation and to subsequently remedy that degradation before system failure, and even before customer detection and complaint. The underlying concept was tabbed, "performance assessment", and was directed towards detection, isolation, and correction of anomaly conditions.

Simultaneously with this effort, the Air Force also proposed investigation into a methodology for automating selected portions of the technical control functions which support the DCS. The intent of this effort was not only to assist in identifying anomaly conditions, but also to facilitate rapid access to information on the health of the DCS and to reduce manual workload.

For nearly ten years, several feasibility and development contractual actions have been pursued, preparatory to the recent contract award to Ford Aerospace and Communications Corporation for the ATEC program. The design of the ATEC equipment, however, does not focus upon the reduction of manpower as a principal objective. Rather its concepts are directed more towards the automation of routine, repetitive, scheduled tasks so as to shift management's attention from diagnostic to remedial actions. An incidental benefit of

the ATEC program is the expected net tri-service manpower reduction of 117 authorizations in technical control and wideband test teams [2].

Air Force communications planners are now pursuing the feasibility and associated benefits of certain unscheduled (and other scheduled) tasks with an eye towards realizing further manpower savings. There appears to be three classes of technical control workload which offer the greatest opportunity for further manpower reductions. These classes of workload are coordination, record-keeping, and reporting, each of which depends upon the effective functioning of an "information-flow management system".

The automation of reporting and record-keeping offers the most significant manpower savings, an additional 106 tri-service authorizations. The necessary functions that would be required of ATEC in order to realize these additional savings are [2]:

1. defining/generating files automatically according to established or changed criteria.

2. generating/altering rules or criteria for building of data files,

3. generating/altering report formats, and

4. assigning rules, criteria, or mathematical operations on stored data for preformatted reports.

These functions are the core of a proposed automatic report generation enhancement to the original ATEC specifications. Unfortunately, this particular enhancement has

been categorized by the ATEC System Project Office as too costly (over $2 million) and too complex (beyond the capability of station processor architecture) to be incorporated into the ATEC program. This enhancement is apparently relegated to a deferred status for follow-on analysis [2].

It appears that presently on-going research and development efforts in the computer science industry, such as microprocessor technology, artificial intelligence, and management information systems, can satisfy the processing requirements of the automatic report generation enhancement in the future. The potential realization of the enhancement, therefore, provides the motivation for this thesis in investigating the applicability of the Rand Corporation developed RITA language.

RITA is a language which allows the user, even a "computer non-sophisticate", to interact with a computer using very English-like commands. It has only one control structure, referred to as production rules, of the form predicate - action (if - then) rules operating on a data base under the control of a rule interpreter/monitor. RITA appears to be an ideal candidate language for implementing the ATEC automatic report generation enhancement.

The next chapter explains the nature of the DCS, focusing on the requirement for automating its technical control functions.

III. DEFENSE COMMUNICATIONS SYSTEM (DCS)

Prior to developing a representative software package for automating the technical control functions of the DCS, it is necessary to have an understanding of the DCS and its inner-workings, component parts, and management structure. Thus, this chapter presents the essential features of the DCS and concludes with explaining the role of system control [3].

A. GENERAL

The DCS is the worldwide complex comprised of long-haul, point-to-point communications facilities, personnel, and material within the Department of Defense. The DCS includes land, sea, and airborne communications required to connect the primary and alternate fixed or mobile command posts of the President, the Secretary of Defense, the Joint Chiefs of Staff, and Unified/Specified Commanders.

The DCS embraces all long-distance point-to-point assets of the three military departments. It includes circuitry by radio (microwave, tropospheric scatter, and high frequency), wire, submarine cable, and satellite from any fixed location to any other fixed location. Excluded from the DCS are the tactical communications systems operated by field commanders. Examples are the Navy's fleet broadcast and ship-to-shore circuits, ground-to-air communications of the Air Force, and

tactical communications of the Army between field units where mobile type equipment is used.

B. FUNCTIONAL RELATIONSHIPS

The Director, Defense Communications Agency (DCA) is responsible to ensure that the DCS is planned, engineered, and operated to meet the long-distance requirements of the DoD and other governmental agencies. To accomplish this, DCA performs the planning, programming, management control and operational direction for the DCS.

Operational direction is the authority to ensure effective operation of the DCS, including directing the operating elements, reallocating DCS operational facilities to accomplish DCA's mission, prescribing standards and procedures, and analyzing system performance and operation of the DCS. This operational direction is applied either directly to DCS TCFs, DCS switching or satellite facilities, or through the appropriate management level of the military services. The DCA Operations Control Complex is the principle agent for exercising operational direction, consisting of 13 operations centers throughout the world.

The military departments are responsible for the operation and maintenance (O&M) of DCS components under their management supervision. Primarily, the Army Strategic Communications Command, The Naval Telecommunications Command, and the Air Force Communications Service (called O&M Managers) carry out these responsibilities through subordinate units

which they command or operationally control. These latter units are generally referred to as operating elements of the DCS. The O&M Managers are also responsible for the budgeting, acquisition, installation, and personnel training aspects in support of the DCS, including leased facilities.

C. SWITCHED NETWORKS AND SUBSYSTEMS

The Automatic Voice Network (AUTOVON) is the principal long-distance, unsecure voice communications network within the DCS. It accommodates voice requirements for essential command and control, operations, intelligence, logistic, diplomatic, and administrative traffic. AUTOVON also provides the interconnection of subscribers in the Automatic Secure Voice Communications (AUTOSEVOCOM) Network and a means for restoration of Automatic Digital Network (AUTODIN) trunks.

The AUTOSEVOCOM provides secure voice service to nearly 1900 special users by means of several types of switches with automatic, semi-automatic, and manual interfaces to terminal instruments.

The AUTODIN functions as a high-speed, computer-controlled, general purpose record communications network. It is an integrated, world-wide network which provides both secure and non-secure data communications. It consists of leased and government-owned Automatic Switching Centers with connecting interswitch trunks.

The Defense Special Security Communications System (DSSCS) provides for exchange of intelligence information

and other record communications which require specialized security procedures. However, much of this traffic is to be satisfied by the AUTODIN II, thereby negating the need for separate switched networks in the future.

Dedicated networks and circuits are decreasing in number as more of them are being integrated into the switched subsystems and networks.

D. TRANSMISSION SUBSYSTEMS

Government-owned submarine cable subsystems include those serving Greenland/Newfoundland, Alaska, Hawaii/Johnston Island, and the Philippine Islands. Most of the government-owned DCS landline cable is in Alaska. Other DCS landline cables are located in Germany and in the Orient.

High frequency (HF) radio operations are being phased down but still provide the primary means of communications in certain areas of the world where there is no other DCS media and where commercial systems do not provide the requisite quality. Areas still being served by HF include the Azores, Iran, Saudi Arabia, and Antarctica. Further, HF is often used as a secondary means of communications in certain parts of the world.

Broadband transmission media is the workhorse of the present DCS. Tropospheric scatter radio, microwave radio, and other line-of-sight radios comprise subsystems that provide communications to US government facilities outside the

continental United States (CONUS). Microwave radio normally provides the most reliable communications. However, microwave requires repeater stations spaced relatively close together which often presents the problem of obtaining host country land rights. Tropospheric scatter radio fills this gap by being able to cover distances up to 500 miles with quality communications. Major concentrations of broadband media are in Europe, as well as much of the rest of the world.

The Defense Satellite Communications System (DSCS) consists of a complex of synchronous satellites which provide nearly total earth coverage with highly reliable communications to fulfill unique and vital national security needs. In addition to fixed ground terminals around the world, this system provides for highly transportable ground, shipboard, and airborne terminals.

DCS communications within the CONUS is provided almost exclusively by leased commercial communications because of economic policy and the high reliability of the service offered. For transoceanic service, the DCS relies heavily upon leased commercial submarine cable and satellite circuitry. In overseas areas, leased commercial communications are used when they provide the quality required and they can be obtained at reasonable costs. Currently, overseas leased commercial communications are used primarily to supplement the government-owned transmission media.

E. SYSTEM CONTROL (SYSCON)

The DCS System Control (SYSCON) incorporates all the means for utilizing DCS assets to insure the optimum DCS performance under all operating conditions. The operating conditions include such vagaries to the system as changing traffic demands, natural or man-made stresses, environmental disturbances, and equipment disruptions. SYSCON is effected through such functional elements as network control, traffic and routing control, performance assessment, status monitoring, and technical control. These functional elements are realized through the SYSCON processes of timely acquisition of performance data, traffic loading status and service quality indications and the rapid analysis, processing and display of this information along with effective execution of control.

There are five hierarchical levels of organizational structure associated with DCS SYSCON. Execution of restoral and reconfiguration actions is accomplished at the lowest levels while analysis and reporting are performed at the higher levels. The relationships between the various SYSCON levels is especially controlled to enhance DCS connectivity, quality and survivability as well as to assure management visibility. Figure 1 depicts the five level hierarchy, along with the level names and functional scope of authority [4].

The TCF is an organic part of a DCS station located at a major transmission node that serves as the point of interface between transmission elements of the DCS and the users. Technical control includes the functions of technical direction,

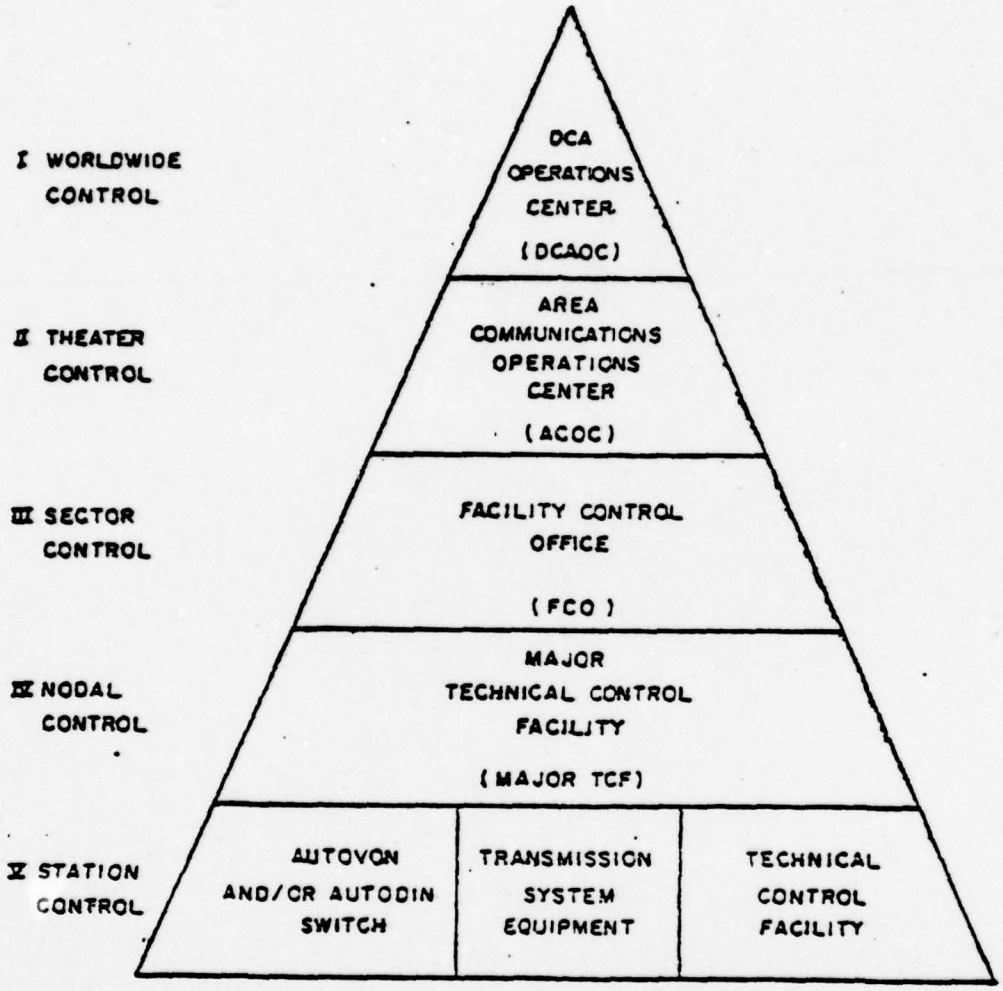


Figure 1. SYSCON HIERARCHY
[adapted from ref. 2]

coordination, communications service restoration, fault isolation, quality control, and status reporting. A TCF is configured so as to enable the technical controller to fully apply the available DCS resources. This capability requires a responsive orderwire and control network between various elements of the DCS.

A TCF is often assigned additional SYSCON responsibilities if it also must function as an Intermediate Control Office or Facility Control Office. These responsibilities involve a broadened scope of data collection/analysis, record-keeping, coordination with affected facilities, and technical management in general.

The tasks of reporting and record-keeping might appear to be a rather straight-forward matter, but upon further investigation several interesting characteristics are observed. Both tasks are essentially manual procedures. The first deals with apprising higher echelons of status information, the second with locally documenting the status on logs and forms. Excessive time (up to 26% of direct workload hours) is spent in reporting, logging and record-keeping. Substantial information is often not recorded because of other pressing workload or merely the manual effort involved. Reporting criteria can be interpreted differently by TCFs, causing inconsistencies in report information. Reports are often received by higher echelons too late to enable the taking of effective actions [2].

The next chapter presents the current efforts underway to automate selected technical control functions, with emphasis on that of record-keeping and report generation.

IV. AUTOMATED TECHNICAL CONTROL (ATEC) PROGRAM

The automation of selected technical control functions is now underway. This chapter describes that automation effort, highlighting the need to also include the tasks of record-keeping and report generation.

A. BACKGROUND

The increasingly complex mission of the DoD dictates ever stringent requirements for highly reliable, responsive communications services. Unfortunately, the operating elements of the DCS have not kept pace with the requirements. As a result, the DCS is too often subjected to unscheduled outages and degraded customer service. A goodly portion of the blame for this situation falls to an inadequate technical control function.

Present TCF procedures and capabilities are insufficient to provide the functions of effective control and direction of communications services. Too often customer service receives the requisite quality control monitoring only after a customer complaint is lodged. The TCF needs to redirect its attention towards, and be provided the tools to realize, a level of technical control that detects system degradations before they adversely affect system performance.

Thus, the concept of an ATEC program was developed - to provide an automated capability that would relieve selected labor-intensive functions performed within the DCS SYSCON.

Specifically, ATEC will provide automation of many of the manual technical control functions.

B. ATEC SYSTEM

The ATEC system will consist of those equipments, personnel, and operating procedures necessary to provide a computer assisted capability for accomplishing many of the transmission and network control functions of the DCS. ATEC will address the processes of circuit and transmission system performance assessment, equipment status monitoring, fault isolation, quality control, centralized sector and nodal system status display, routing plans and record storage, and report preparation.

The ATEC subsystems will parallel the DCS SYSCON levels as shown in figure 1. Operational direction and control actions generally flow downward in the hierarchical structure while reports flow upward. The organizational levels of the DCS and ATEC correspond approximately to geographic boundaries, with several DCS stations being controlled by a nodal control and several nodal control points by a sector control. The ATEC components will be superimposed over the DCS according to the configuration requirements of each facility, properly sized and interconnected. Only the sectors, nodes, and stations will receive ATEC components which will normally operate as a system; nodal and station controls will have a degraded mode capability to operate as stand-alone facilities in the event of failure of interconnections [3].

The overall resultant of implementing the ATEC system is expected to be improved effectiveness and efficiency in the management and technical control of the DCS. As such, ATEC will become a subset of DCS SYSCON, satisfying the requirements of transmission control and performance assessment. In fact, ATEC will be a prime contributor to the DCA's efforts in developing an integrated, automated SYSCON capability to manage the entire DCS.

The process of acquiring an ATEC system was initiated in the late 1960's by the US Air Force, sponsored by DCA. Early efforts began with a research, development, test, and evaluation program to develop specialized test equipment, computer memory and processing, and various software algorithms. In the mid-1970's, the procurement approach was revised from using design specifications to functional performance specifications. This latter action was taken to achieve a more competitive procurement and reduced acquisition costs, as well as take advantage of technical developments, especially for greater modularity and flexibility.

Since changing to a functionally oriented test program, test and evaluation efforts have been intensified. A joint test team was organized to evaluate the performance of a test bed system that had been established in Europe. Preliminary test data indicated questionable benefits of manpower savings and improved DCS performance. A subsequent cost benefits analysis of the ATEC program has vindicated the acquisition program.

The major conclusion of the cost benefits analysis is the significant manpower savings associated with the ATEC program. As presently envisioned, ATEC would facilitate a 30% reduction in TCF manning. After allowing for manpower increases to support the ATEC system itself, manpower savings would amount to approximately 89 authorizations. Because of improved performance assessment, it is expected that 28 manpower spaces could be eliminated from the current force of DCA wideband evaluation teams. Thus, net tri-service manpower reductions are estimated conservatively at 117 billets [2].

The other benefits of the ATEC program also support continuation of the acquisition. ATEC is the basis for the automated DCS SYSCON and is endorsed by the DCS Five Year Program, FY 1980-1984. There is no viable alternative to ATEC currently under consideration. Because of its modular nature for implementation, ATEC will be able to accommodate even those significant portions of the DCS which are non-standardized as well as those of an evolving digital DCS.

The acquisition of ATEC will attempt a phased procurement and delivery of components. A request for proposal and functional specifications for a Low Rate Initial Production (LRIP) system were released to industry. Ford Aerospace and Communications Corporation (FACC) has been awarded the contract for these LRIP components and has developed their technical proposal to an extensive degree.

Subject to final System Design Review, FACC tentatively proposes to produce applications programs written in a single higher order language (HOL), FORTRAN IV-PLUS. The use of MACRO-11 assembly language will be restricted to those programs where using a HOL would not be efficient or practical. The Digital Equipment Corporation will be the prime equipment source for PDP-11/70 minicomputers (512 kbit memory) for sector and nodal levels and LSI-11/2 microprocessor hardware for station levels. The RSX-11M operating system and the CODASYL DBMS-11 data base management system will be used. The operator query capability for examining the data base will be accomplished using a FACC provided language called QUERY [7].

C. RECORD-KEEPING AND REPORTS

The ATEC baseline requirements include the need for providing computer assistance in report preparation and local record-keeping. This computer assistance is to be used for displaying a variety of different fixed report formats to the controller, for allowing the entry by either machine or controller of address information and data items into the report formats and for automatically transmitting and obtaining hard copy of completed reports.

The ATEC man-machine communications interface is to allow the use of conventional technical control terminology, units, and mnemonics for entry of information and receipt of response. The machine should complete or acknowledge

requests for information or message delivery from the operator in a timely manner.

Messages and reports between controller terminals will be preformatted to the greatest extent possible, with unique identifiers for originating station, date-time group, and other pertinent information to identify the message content. Controllers will be able to send multiple address messages simultaneously to receiving destinations [3].

While it would appear that the above descriptions are adequate for specifying the needed function of record-keeping and report generation, the LRIP specifications do not sufficiently address the requirement. The contractually binding portions of the specifications are unclear and leave much of the requirement open to interpretation [2].

Further, the original emphasis of the ATEC program was towards the automation of routine, repetitive tasks which would directly sustain DCS high quality performance. Reduction in TCF manning was only a secondary benefit. There is now a high level of interest at Hq Air Force Communications Service (AFCS) in potential manpower reductions resulting from ATEC, to the degree that a significant effort is underway there to append the original LRIP specifications with proposed enhancements to strengthen and more fully define them [2].

The enhancement related to automatic report generation has been evaluated by the ATEC System Project Office and determined to be both too costly and complex for inclusion

with the LRIP specifications. Unfortunately, this enhancement would have yielded an additional manpower savings of 106 authorizations. Needless to say, the communications planners are quite frustrated with the prospect of this enhancement being merely deferred for future follow-on analysis [2].

D. AUTOMATIC REPORT GENERATION ENHANCEMENT

The enhancement provides for technical controllers at sectors, nodes, and stations having the capability to specify existing ATEC data to be collected, to specify the reports to be prepared from the collected data, and to invoke the preparation of reports. It requires the following functions:

1. automatic entry of data into reporting forms,
2. automatic generation of files from which records and reports are generated,
3. capability to alter report formats,
4. capability to alter rules for file building and report generation,
5. algorithms for automated statistical analysis, and
6. information-flow management of work-in-progress.

This enhancement would be effected by technical controllers by using two related but different modes of operation - a setup mode and an operate mode. During the setup mode, the operator (a programmer or perhaps even a technical controller) would establish the basic parameters to control

rule-making and file manipulation. The operate mode would then allow the technical controller to use and even modify those pre-established parameters to actually generate records and reports.

The dialogue expected between the operator/technical controller and the machine is visualized to be similar to those type commands normally utilized for file management, text editing, and data manipulation at an on-line interactive terminal in a time-shared environment, rather than a traditional high order language for computer programming.

The main thrust of this thesis is to develop representative programs which will facilitate the required functions of this enhancement. The next chapter describes the rudimentary features of a candidate ATEC language for this automatic report generation enhancement.

V. RULE-DIRECTED INTERACTIVE TRANSACTION AGENT (RITA)

This chapter will describe an innovative approach to computer programming, termed production rule systems and in particular the RITA system. Because of the unique features of RITA, it offers a potential software solution to the ATEC data management requirement.

A. PRODUCTION RULE SYSTEMS

Typical computer software programs operate in a predetermined, fixed sequence process, knowing which coded commands are to be next executed in the program. This class of programs has been of tremendous utility to a wide range of users. However, for certain applications a different approach in program control is needed, i.e., quite often it is desirable to have the data base contents determine the program execution. Thus was born the development of production rule systems.

A production rule system is a program comprised of a set of statements or rules which are of the following form: if a particular condition exists, then perform a certain action. Conventional shorthand notation for the production rule form is condition → action. The condition is a statement(s) about the contents of the data base. The action is a process(es) which is performed upon the contents of the data base. The rules operate upon the data base under the control of a monitor which interprets the rules.

When the program is executed, the data base is checked by the production rule system to determine if the first rule condition(s) is true. If it is found to be true, then the associated action(s) is performed upon the data base. After the rule has successfully "fired" or if the rule condition(s) is not true, the next rule is similarly checked. This process of checking the rule condition(s) and performing the action(s) is recycled continuously until all rules are found to be false or a special halting function is encountered. In this manner, the production rules cause the program execution to be dependent upon the contents of a continuously modified data base.

The conventional production rule systems described above are termed condition-driven since the rules make contact with the data base through the rule conditions. Another class of production rule systems interact with the data base through the rule actions, and thus are termed action-driven. This second class of systems attempt to prove the action by first checking for it directly in the data base or failing in that, by showing that the associated conditions are true, in a deductive inference fashion [12].

A number of different applications have used either the condition-driven or action-driven systems. The next section, however, will present a unique development which utilizes both condition- and action-driven production rule systems.

B. DEVELOPMENT OF RITA

RITA is an acronym which originally stood for Rand Intelligent Terminal Agent but now has come to be know as Rule-directed Interactive Transaction Agent. The RITA system was designed and implemented in the mid-1970's by the Information Sciences Department of Rand Corporation under development contract from the Information Processing Techniques Office of the Advanced Research Projects Agency (ARPA) of the DoD. RITA is intended to serve as a stand-alone resource for the purposes of text manipulation, routine interaction with local and remote information/computing systems and networks, or even development of heuristic models to aid in decision-making and analysis [8,9].

The design philosophy of RITA is based upon the following premises:

1. Future users of computer systems will tend to be from outside the realm of "computer sophisticates". These users will need computer systems designed to their level of expertise.

2. Microprocessor technology advances are projected to make available in the mid-1980's interactive, intelligent terminals which have the processing capability of today's minicomputers.

3. This user terminal should have sufficient data storage and handling capability to allow stand-alone operation, immediacy of response, and textual editing and retrieval.

4. The agent must provide two-way communications with external computer systems and with the user, especially in explaining its behavior to the user.

5. The agent must maintain a catalogue of assigned tasks, schedules, and deadlines which are protected during system downtime and managed according to its own calendar and clock time functions.

The RITA system is a collection of computer programs designed to facilitate the writing of user application programs, called user agents, which provide the user with an intelligent interface with the external computer system. The RITA programs are written in the C language, with a few small subroutines written in assembly. The present RITA Version 2.0 runs under the UNIX Version 6 operating system, with Rand modifications, and often makes use of many of the UNIX system capabilities. RITA has been human-engineered to the extent that the user agents can be written in near-English language [9].

The RITA system is very large and depends upon the use of separate instruction and data registers if a minicomputer, such as the Digital Equipment Corporation PDP 11/45 or 11/70 presently in-service at Rand, is used. The current Rand system can accommodate approximately 280 kbytes of program and data storage in core. This capacity would be equivalent to about 120 rules operating upon a data base of 130 objects. It is this instruction set and data base that will eventually be able to reside in future intelligent terminals [9].

The RITA system software is available to government contractors and agencies and private organizations for a modest charge to cover tape and machine costs at Rand. Rand retains a corporation official to serve as the RITA Interface (RIF) to discuss matters of distribution, documentation, bugs, etc. Bolt Beranek and Newman (BBN) is responsible for maintaining the RITA system (based on UNIX) within the context of the ARPA program. A number of computer centers maintain the RITA system on-line, while others have the distribution tapes on hand [10].

C. UTILIZATION OF RITA

Learning the proper commands, processes, syntax, and unique features of RITA is a special study in itself. The reader is referred to the source documents listed in the bibliography for a detailed study of RITA [8,11]. The following description is offered as only a rudimentary overview.

A RITA program is a set of commands along with a collection of rules and/or goals which can be tested and which direct certain actions if the rule premises are true. The data which are generated or acted upon are stored in the form of objects. The data base consists of a collection of object-attribute-value triples.

RITA is usually run in an interactive mode during the creation of the rule-sets and initialization of the data objects. As the user creates objects and rules, RITA checks

them for syntax errors and then stores the input. There can be considerable dialogue between the user and RITA during the course of creating, viewing, and storing these items. RITA types an asterisk (*) to the user whenever it is ready to receive more input. All input must be terminated by a semicolon (;) and carriage return.

The user can invoke the RITA program to run and thus test all entered rules, executing those with true premises. Further, all entered rules and data can be stored for subsequent recall and utilization. RITA can operate either with input from the user or from previously stored files without further human intervention.

RITA objects are stored in the form of an object class or type which has an arbitrary number of attributes, each of which has an assigned value. These object-attribute-value triples are operated upon by the rule sets; their ability to be readily changed in attribute and value gives the RITA process its dynamic nature. Thus, the content of the data base can have just as profound affect on the program output as the rules and commands. An example of a RITA object is:

```
*object horse
    color is "grey",
    age is "7",
    owner is ("linda","mark");
```

A RITA command may be either a rule or goal, an action by itself, or a directive to the system.

RITA rules are the condition-action rule sets discussed earlier. The "if" part of a rule consists of any number of

premises, all of which must be true in order for the rule to "fire". The premises test various attributes of the data base (the objects), and the resulting actions can then modify the data base. The actions can also have affect on other systems external to RITA, such as the UNIX data base or certain ARPANET processes. An example of a RITA rule is:

```
*rule airforce
   if the service of the man is "USAF"
   then create a uniform whose color is "blue";
```

A RITA rule will continue to "fire" until either the premises are no longer true or a special halting function is encountered. The halting function is used by inserting as the last action in a rule the phrase, "and return success;". This causes not only the program to halt but to also print out to the user, "Success!", at the end of the program output.

RITA goals are special kinds of rules used in deductions. With few exceptions, all premises which can be used for rules can be used for goals. When a goal is executed, the one relevant action clause is "fired" which chains it to the deduction. An example is:

```
*goal graduation
    if the thesis of the student if "complete"
    then set the status of the wife to "glad";
```

When a RITA deduce command is enacted, RITA first searches all goals to find the appropriate action which would yield the desired result. Then the selected goal is

tested for whether the premises are true. This process usually leads to testing other goals in a "backward chaining process" until a true premise is discovered.

RITA commands can perform several other actions on the data base, such as replace or remove values, send data to another data base, print specified objects and/or rules, and load additional files. RITA directives issue orders to the RITA system, such as run a program, debug a program, establish a temporary access to UNIX, and exit the program.

RITA also contains several built-in functions to ease the desired processing of data, comparable to simplified subroutines. Further, fundamental arithmetic operations can be called up by use of the following symbols: + - / * .

The UNIX operating system provides a pipe capability which RITA can use to establish communications channels, called UNIX ports, with external systems. This capability allows RITA to send/receive messages to/from the user or itself. Further, it enables a computer-to-computer link with external systems using ARPANET procedures.

This concludes the description of RITA. The next chapter shows a representative RITA agent - rules, objects, and commands - that could be used to implement the ATEC automatic report generator enhancement.

VI. RITA AGENT FOR ATEC

This chapter presents in a sequential fashion, a methodology that could be used to program a RITA agent for producing the ATEC automatic report generator. The agent was built piece-meal, adding on rules incrementally to demonstrate the fundamental requirements of the ATEC enhancement.

A. A REPORT FORMAT

The format for a report, which could be established by the set-up technical controller, used a single rule which queried the user, or operating technical controller, for certain data values. These values were then assigned to certain attributes and the resulting report with data value entries were displayed to the operator.

The rule for doing this was:

```
rule report
if status of report is "due"
then set the name of report to "Trouble Ticket"
and send "What is the CCSD?" to user
and receive next {anything 'ccsd' followed by "^j"} from user
and set ccsd of report to 'ccsd'
and send "What is the source of the report?" to user
and receive next {anything 'source' followed by "^j"} from
    user
and set source of report to 'source'
and send "What is circuit priority?" to user
and receive next {anything 'priority' followed by "^j"} from
    user
and set priority of report to 'priority'
and send "What is your station?" to user
and receive next {anything 'station' followed by "^j"} from
    user
and set station of report to 'station'
and send "Is the circuit send or receive?" to user
and receive next {anything 'snd/rcv' followed by "^j"} from
    user
```

```
and set snd/rcv of report to 'send/rcv'
and send "Enter the date-time-group of the outage, in the
    form (last digit of calendar year, julian date, zulu
    time)." to user
and receive next {anything 'dtg-out' followed by "^j"} from
    user
and set dtg-out of report to 'dtg-out'
and send "Enter the date-time-group of restoral, in the form
    (last digit of calendar year, julian date, zulu time."
    to user
and receive next {anything 'dtg-in' followed by "^j"} from
    user
and set dtg-in of report to 'dtg-in'
and send "What is the code for the reason for outage?" to
    user
and receive next  anything 'rfo-code' followed by " j"  from
    user
and set rfo-code of report to 'rfo-code'
and send "Enter any desired remarks, limited to one line."
    to user
and receive next {anything 'remarks' followed by "^j"} from
    user
and set remarks of report to 'remarks'
and send concat("^j^j^j^j^j") to user
and display object report
and set status of report to "not-due"
and set status of techcon to "working4"
and return success;
```

As each question was sent to the user, the program stopped until the response was typed at the keyboard. A typical output from running this rule was a trouble ticket report as follows:

```
What is the CCSD?
JJAV9ABC
What is the source of the report?
Maintenance section
What is circuit priority?
02
What is your station?
MRE
Is the circuit send or receive?
send
Enter the date-time-group of the outage, in the form (last
    digit of calendar year, julian date, zulu time).
(8,345,2100)
Enter the date-time-group of restoral, in the form (Last digit
    of calendar year, julian date, zulu time).
(9,346,1545)
What is the code for the reason for outage?
23
Enter any desired remarks, limited to one line.
Bad resistor located in transmitter and is now on order.
```

```
OBJECT report<1>:
        status     IS    "due",
        name       IS    "Trouble Ticket",
        ccsd       IS    "JJAV9ABC",
        source     IS    "Maintenance section",
        prior ty   IS    "02",
        station    IS    "MRE",
        snd/rcv    IS    "send",
        dtg-out    IS    "(8,345,2100)",
        dtg-in     IS    "(9,346,1545)",
        rfo-code   IS    "23",
        remarks    IS    "Bad resistor located in transmitter
                         and is now on order.";
```

Repeated running of the rule creates additional versions of the object "report" which are distinguished by local labels assigned automatically by RITA. These object names would be of the form:

```
OBJECT report<1>
OBJECT report<2>
   :     :
OBJECT report<n>
```

where n is the number of repeated runnings of the rule.

The format of the printout was that typically encountered when displaying an object. If a different format were desired, then a special "display concat" command could have been devised. One such version and the resulting printout was:

```
and display concat("CCSD:  ",ccsd of report,
^jSource:  ", source of report,"^j Priority:  ",
priority of report,"^j Snd/rcv:  ",
snd/rcv of report,"^jDTG-OUT:  ",dtg-out of report,"^j
DTG-IN:  ",dtg-in of report,"^jRFO Code:  ",rfo-code
of report,"^jRemarks:  ",remarks of report
"^j^j^j^j") and return success;
```

TROUBLE TICKET

CCSD: JJAV9ABC
Source: Maintenance section
Priority: 02
Station: MRE
Snd/rcv: send
DTG-OUT: (9,345,2100)
DTG-IN: (9,346,1545)
RFO Code: 23
Remarks: Bad resistor located in transmitter and is now on order.

Unfortunately, it was not possible to store and then manipulate this printout as a RITA object. Creating a printout format different than that produced by RITA for an object was quite undesirable. Only RITA objects were used for report printouts in the remainder of this program.

B. CHANGING REPORT ENTRIES

There were several options available to the user to modify report entries. One way was to revise the basic rule itself, using normal UNIX text editing procedures, and change the content of the questions. Another was to rearrange the order of the send and receive action pairs to revise the sequence of the attributes in the printout.

If only changes in values were desired, the user could have deleted the object report and run the rule again. Or individual value entries could have been changed by substituting another value using a command such as:

*set the ccsd of the report to "xxxxxxxx";

C. STORING AND DELETING THE REPORT

Storing a RITA object was quite straight-forward. The simple command which follows would store the report in a UNIX file named "trubltik":

```
*display object report to file trubltik;
```

The external UNIX files could be examined to verify that the report had indeed been stored by entering,

```
*shell;
%ls
%cat trubltik
```

The "shell" command caused RITA to temporarily leave the RITA mode and enter UNIX. The "ls" and "cat trubltik" commands caused, respectively, the index of all UNIX files and the contents of file "trubltik" to be displayed to the user.

This new file, "trubltik", was then permanently filed until it was purposely removed from the UNIX external files. There were two methods for removing UNIX Files. One method used the universal UNIX command,

```
%rm trubltik
```

The other method was a Rand developed procedure which used the special UNIX command,

```
%del trubltik
```

This second form was preferred since before removing the file it repeated the filename to be deleted and then asked the user to verify that the file was actually to be removed. Thus, two positive actions were required to delete a file, whose inadvertent deletion could possibly have been disastrous.

D. SENDING A REPORT

In order for RITA to send information outside of itself, it had to activate a communications link, called a UNIX port, to the external location. Several protocols needed to be established with the distant computer facility to verify the user, password, and account, as well as to transfer the file and to coordinate end of transfer and disconnect.

The basic provisions of the ARPANET File Transfer Protocol (FTP) satisfied the above requirements. The following program established the link, displayed its progress, printed the index of filenames at the remote location before and after the file was transferred to the distant computer, and then disestablished the link.

```
object send.report:
state is "start";

object portl:;

rule ftp:
if: the state of send.report is "start"
then: send concat("ftp", "^j") to portl
and receive next {"Host:" followed by anything} for 15 seconds
from portl as the response of port 1
and set state of send.report to "sendhostname";
```

```
rule sendhostname:
if: the state of send.report is "sendhostname"
and the response of portl is known
then: send concat("isia", "^j") to portl and receive next
{anything followed by
"Connections established" followed by anything} for 15 seconds
from portl as the response of portl
and display concat ("^j^j", response of portl)
and set state of send.report to "senduser";

rule senduser:
if: the state of send.report is "senduser" and
the response of portl is known
then send concat("user nps-accat", "^j") to portl
and receive next {anything followed by "User name accepted"
followed by anything} for 15 seconds from portl as response
of port 1
and set state of send.report to "sendpass";

rule sendpass:
if: the state of send.report is "sendpass"
and the response of portl is known
then send concat("pass", "^j") to portl
and receive next {"Password:"} for 15 seconds from portl
as response of port 1
and set state of send.report to "sendpassword";

rule sendpassword:
if the state of send.report is "sendpassword"
and the response of portl is known
then send concat("c77", "^j") to portl
and receive next {anything followed by "Login completed"
followed by anything} for 15 seconds from portl as response
of portl
and display concat("^j^j", response of portl, "^j^j")
and set state of send.report to "sendacct";

rule sendacct:
if:  the state of send.report is "sendacct"
and the response of portl is known
then send concat("acct", "^j") to portl
and receive next {anything followed by "Account #"
followed by anything} for 15 seconds from portl as the
response of portl
and set state of send.report to "sendacctcode";

rule sendacctcode:
if: the state of send.report is "sendacctcode"
and the response of portl is known
then send concat("cc-", "^j") to portl
and receive next {anything followed by "Account command
```

```
accepted" followed by anything} for 15 seconds from portl
as response of portl
and set state of send.report to "logged in";

rule directory1:
if: the state of send.report is "logged in"
then: send concat("status", "^j") to portl
and receive next {anything followed by "NPS-ACCAT" followed
by anything followed by "ARCHIVE" followed by anything}
for 60 seconds from portl as the response of portl
and display object portl
and set the state of send.report to "listing1";

rule sendfile:
if: the state of send.report is "listing1"
and the response of portl is known
then: send concat("store trubltik trubltik", "^j") to portl
and receive next {"Transfer completed"} from portl as
response of portl
and display concat ("^j^j", response of portl, "^j^j")
and set state of send.report to "transferred";

rule directory2:
if: the state of send.report is "transferred"
then: send concat("status", "^j") to portl
and receive next {anything followed by "NPS-ACCAT" followed
by anything followed by "ARCHIVE" followed by anything}
for 60 seconds from portl as the response of portl
and display object portl
and set the state of send.report to "listing2";

rule logout:
if: the state of send.report is "listing2"
then: send concat("bye", "^j") to portl
and receive next {anything followed by "BYE command received"
followed by anything} from portl as the response of portl
and set state of send.report to "logged out"
and display concat ("^j^j", response of portl, "^j^j")
and set status of techcon to "working7"
and return success;
```

The output displayed to the user during the running of this program was,

```
Unknown host name
Host: Connections established" "

230 Login completed
```

```
"OBJECT port1<1>:
        response        IS    ".
> 151 <NPS-ACCAT>
151 $XED-MODE-FILE$.;1
151 TRUBLTIKFILE.;1
151^v^+]ARCHIVE";

 "

-DIRECTORY[.;1
200 End of status.
> > 200 Allocation ignored on this system.
255 SOCK 3276899858
250 Store of <NPS-ACCAT>TRUBLTIK.;1;P775200;ACC-, ASCII
    type, started.
252 Transfer completed

"OBJECT port1<1>:
        response        IS    ".
> > 151 <NPS-ACCAT>
151 $XED-MODE-FILE$.;1
151 TRUBLTIK.;1
151 TRUBLTIKFILE.;1
151 ^v^+]ARCHIVE";

 "

-DIRECTORY[.;1
200 End of status.
    % % 231 BYE command received
```

An operational disadvantage of this portion of the program was that security of the system could be compromised in the process of declaring the password and account code. Further, additional program rules were needed to accommodate a failure in establishing the communications link or transferring the data.

E. RECEIVING REPORTS

The ATEC program is expected to generate many reports flowing between technical control facilities, which would have to be sorted, catalogued, stored and analyzed. RITA could provide this capability, receiving reports as they

are generated and sent in to the user or by querying the distant location periodically.

The next RITA program was very similar to the earlier one for establishing an FTP interconnect. The first seven rules were essentially the same and are omitted to save space. The last four rules were replaced with the following:

```
rule recvfile
if the state of send.reports is "logged in"
and the response of port2 is known
then send concat ("retrieve trubltikfile trubltikfile",
"^j") to port2
and receive next {anything followed by "Transfer completed"
followed by anything} from port2 as response of port2
and display concat ("^j^j", response of port2)
and set state of send.reports to "transferred";

rule logouts:
if: the state of send.reports is "transferred"
then: send concat("bye", "^j") to port2
and receive next {anything followed by "BYE command received"
followed by anything} from port2 as the response of port2
and set state of send.reports to "logged out"
and display concat ("^j^j", response of port2, "^j^j")
and set status of techcon to "working9"
and return success;
```

The output of the program was the following:

```
Unknown host name
Host: Connections established" "

230 Login completed

" "

.
> 255 SOCK 3276899859
250 ASCII retrieve of <NPS-ACCAT>TRUBLTIKFILE.;3 started.
252 Transfer completed" "

.
> > 231 BYE command received

"Success!
```

This program did nearly the same as the earlier FTP program, except that the file, "trubltikfile", was retrieved rather than one being sent and the new file was not displayed until the following command was sent:

```
*shell;
%cat trubltikfile
```

The new file contained a collection of trouble ticket reports, in an abbreviated form to conserve space and modified to allow RITA mathematical processing. Notice that the second and fourth reports contain a different CCSD value than the others. It contained the following reports:

```
object report
status is "due",
ccsd is "JJAV9ABC",
dtg-in is ("9","362","22","15"),
dtg-out is ("9","361","09", "00");

object report
status is "due",
ccsd is "GGSD89JK",
dtg-in is ("9","356","23","15"),
dtg-out is ("9","355","12","55");

object report
status is "due",
ccsd is"JJAV9ABC",
dtg-in is ("9","345","15","00"),
dtg-out is ("9","344","10","10");

object report
status is "due",
ccsd is "CVFG45JK",
dtg-in is ("9","333","18","50"),
dtg-out is ("9","332","15","35");

object report
status is "due",
ccsd is "JJAV9ABC",
dtg-in is ("9","322","03","00"),
dtg-out is ("9","322","01","10");
```

```
object report
status is "due",
ccsd is "JJAV9ABC",
dtg-in is ("9","289","20","00"),
dtg-out is ("9","279","21","05");

object report
status is "due",
ccsd is "JJAV9ABC",
dtg-in is ("9","267","15","15"),
dtg-out is ("9","266","04","40");

object report
status is "due",
ccsd is "JJAV9ABC",
dtg-in is ("9","237","23","40"),
dtg-out is ("9","235","02","35");

object report
status is "due",
ccsd is "JJAV9ABC",
dtg-in is ("9","195","01","20"),
dtg-out is ("9","194","20","55");

object report
status is "due",
ccsd is "JJAV9ABC",
dtg-in is ("9","001","13","40"),
dtg-out is ("8","364","18","45");
```

F. STATISTICAL ANALYSIS

The ATEC report generator enhancement also requires a certain degree of statistical manipulation of data parameters. The following program demonstrated RITA's capability in this area, using the "trubltikfile" data base as an example for calculating the mean and variance values for circuit downtimes.

The program first initialized the data objects and then screened through the outage reports sorting for the desired circuit outages, in this case for CCSD JJAV9ABC. Set

ordered was specified in order for the rules to cycle individually through all objects before the next rule was tested.

```
object statistic
status is "compute",
denominator is "0",
numerator is "0",
summation is "0";

object downtime;

set ordered;

rule statsl
if there is a report whose status is "due"
and ccsd of report is not "JJAV9ABC"
then set status of report to "not-applicable";
```

The next set of rules tested the values of minutes, hours, and days in the in- and out-of-service date-time-groups and prepared those values for a subtraction process.

```
rule stats2
if there is a report whose status is "due"
and member 4 of dtg-in of report is less than member 4 of
dtg-out of report
then put <member 4 of dtg-in of report + 60> into dtg-in
of report after member 3
and put <member 3 of dtg-in of report - 1> into dtg-in of
report after member 2
and remove member 6 from dtg-in of report
and remove member 4 from dtg-in of report
and set status of report to "minutes";

rule stats3
if there is a report whose statis is "due"
then set status of report to "minutes";

rule stats4
if there is a report whose status is "minutes"
and member 3 of dtg-in of report is less than member 3 of
dtg-out of report
then put <member 3 of dtg-in of report + 24> into dtg-in
of report after member 2
```

```
and put <member 2 of dtg-in of report - 1> into dtg-in of
report after member 1
and remove member 5 from dtg-in of report
and remove member 3 from dtg-in of report
and set status of report to "hours";

rule stats5
if there is a report whose status is "minutes"
then set status of report to "hours";

rules stats6
if there is a report whose status is "hours"
and member 2 of dtg-in of report is less than member 2 of
dtg-out of report
then put <member 2 of dtg-in of report + 365> into dtg-in
of report after member 1
and set status of report to "days"
and remove member 3 from dtg-in of report;

rule stats7
if there is a report whose status is "hours"
then set status of report to "days";
```

The next rule sets performed the subtractions, converted the times to minutes, and performed the division for determining the mean of the circuit downtimes (in minutes).

```
rule stats8
if there is a report whose statis is "days"
then set the downtime of report to 1440 * <member 2 of dtg-in
of report - member 2 of dtg-out of report> + 60 * <member 3
of dtg-in of report - member 3 of dtg-out of report> +
<member 4 of dtg-in of report - member 4 of dtg-out of report>
and set status of report to "downtime"
and put the downtime of report into downtimes of statistic;

rule stats9
if there is a report whose status is "downtime"
then set the denominator of statistic to <denominator of
statistic + 1>
and set the numerator of statistic to <numerator of statistic
+ downtime of report>
and set ccsd of downtime to ccsd of report
and set the status of report to "processed";
```

```
rule statsl0
if the status of statistic is "compute"
then set the quotient of statistic to <numerator of statistic/
denominator of statistic)>
and set the status of statistic to "quotient";
```

The next two rules calculated the variance of the circuit downtimes.

```
rule statsl1
if there is a report whose status is "processed"
then set summation of statistic to <summation of statistic
+ <<ABS(downtime of report - quotient of statistic)> *
<ABS(downtime of report - quotient of statistic)>>
and set status of report to "finished";

rule statsl2
if the status of statistic is "quotient"
then set variance of downtime to concat(<summation of
statistic/denominator of statistic>," minutes squared")
and set status of statistic to "variance";
```

At this point it was desired to compute the standard deviation by taking the square root of the variance. Unfortunately, RITA did not have a built-in function for computing square root.

Then the program converted the mean back to days, hours, and minutes and finally printed the results.

```
rule statsl3
if the status of statistic is "variance"
and quotient of statistic is less than 60
then set mean of downtime to concat(quotient of statistic,
" minutes")
and set status of statistic to "computed"
and display object downtime
and set status of techcon to "workingll"
and return success;

rule statsl4
if the status of statistic is "variance"
and quotient of statistic is greater than 60
and quotient of statistic is less than 1440
```

```
then set mean of downtime to concat(FLOOR(quotient of
statistic/60," hours, ",MOD(quotient of statistic,60),
"minutes")
and set status of statistic to "computed"
and display object downtime
and set status of techcon to "working11"
and return success;

rule stats15
if the status of statistic is "variance"
and quotient of statistic is greater than 1440
then set days of statistic to FLOOR(quotient of statistic/
1440)
and set hours of statistic to FLOOR(MOD(quotient of
statistic,1440) / 60)
and set minutes of statistic to MOD(MOD(quotient of
statistic,1440),60)
and set mean of downtime to concat(days of statistic," days, ",
hours of statistic," hours, ",minutes of statistic,"minutes")
and set status of statistic to "computed"
and display object downtime
and set status of techcon to "working11"
and return success;
```

The program output was:

```
OBJECT downtime 1 :
    ccsd      IS   "JJAV9ABC",
    variance  IS   "18419354.686 minutes squared",
    mean      IS   "2 days, 9 hours, 13.75 minutes";

Success!
```

The next chapter presents another RITA agent which operates in an interactive mode with the user to present all the rules and commands discussed in this chapter.

VII. INTERACTIVE AGENT

In this chapter an interactive, user-prompting agent is explained that would allow the user to execute the rules, commands, and objects presented in the previous chapter.

A. INTRODUCTION

The interactive routine explained in this chapter allowed a terminal operator untrained in RITA programming to experience the power and versatility of RITA. The routine consisted of a series of rules that issued narrative explanation to the user of those keyboard entries needed to generate a report, change format, send and receive files, and obtain statistical data.

The operator input to the agent was a series of RITA commands to load files of data base objects and rules and to execute those rules. The ultimate output of the agent was the products resulting from the rule executions.

At the time of the writing of this thesis, the complete program was contained in the following UNIX filenames:

demo
duplicate
recv.report
rulereport
stats
xmit.report

which were stored at host 199, RAND-UNIX on the ARPANET. Readers who desire to obtain these files should contact Dr. Gary Poock of the Naval Postgraduate School (see distribution list of the thesis).

B. EXPLANATION OF AGENT

The first three rules of the agent provided general information on the purpose of the program and procedural matters. They could have comprised a single rule except that there was a RITA limit for the length of a string of characters that could be displayed on a user's terminal.

```
rule one
if the status of the techcon is "ready"
then send "^J^J
```
This program demonstrates a capability for using RITA programming techniques to produce an automatic report generator enhancement for the ATEC program. It allows the technical controller to perform the following functions:

A. Define/generate data files automatically according to rules established by the technical controller,

B. Generate/alter rules for building data files.

C. Generate/alter report formats, and

D. Assign rules or mathematical operations on stored data." To user;

```
rule two
if the status of the techcon is "ready"
then send "^J^J
```
Periodically during this demonstration, it will be necessary for you to enter certain commands at the keyboard. You will be notified of what to type. The exact words will be displayed and should be typed in lower case precisely as shown, without the prompt symbol. The carriage return should be typed whenever the symbol, <cr>, is shown." To user;

```
rule three
if the status of the techcon is "ready"
then send "^J^J
```
Should you make a typing error, correct it on the same line using the '#' key to erase each erroneous key stroke, before you type the carriage return. If you discover your error after typing the carriage return, then you must start the demonstration over from the beginning by doing the following:

A. While holding down on the 'ctrl' key, press the 'esc' key. Then release both keys and press the carriage return key. This should cause the terminal to type out the RITA prompt (*).

B. Start the demonstration over by typing the following on separate lines after receiving either the RITA prompt (*) or UNIX prompt (%), as shown below:

```
*exit;    <cr>
%rita     <cr>
*load demo;     <cr>
*run;     <cr>
```

These same procedures shown above must be followed in the event that either an error or failure message is displayed by the terminal." To user;

Rule four showed how to temporarily disconnect from RITA by creating a UNIX shell program which displayed the interactive rule that generated a trouble ticket report.

```
rule four
if the status of the techcon is "ready"
then send "^J^J
```
First, the demo will show the set-up mode for creating a report format. In this case, a trouble ticket format is prepared that will automatically receive value entries specified by the technical controller. When the RITA prompt (*) appears, please type,

```
*shell;          <cr>
%cat rulereport      <cr>
```

When the terminal stops typing, then hold down the 'ctrl' key while you press the 'd' key (once only) which should cause the RITA prompt to appear. Then type the following,

```
*run;    <cr>
```

```
in order for the program to continue."  To user
and set the status of techcon to "working1"
and return success;
```

Rule five explained how to modify the report format and showed how to load this report generator rule into the RITA session. The latter action also caused a syntax error check to be performed.

```
rule five
if the status of the techcon is "working1"
then send "^J^J
```

The above UNIX file is an example of what the set-up operator would produce to create a new report format. Normal UNIX text editing procedures could then be used to modify the entries of format. Once the format is finally decided upon, then the file is loaded into the RITA system.

Loading a file into the RITA system causes the file to be carefully checked to ensure that the rules and objects have been correctly written. Any errors will be indicated for the user to make corrections. Once all corrections have been made and the file is again loaded into RITA, the system will respond with which rules and objects have been added (to this RITA session in progress) and finish with the RITA prompt.

To see this action, please type,

```
*load rulereport;     <cr>
```

When the terminal stops typing, then type,

```
*run; <cr>
```

```
in order for the program to continue."  To user
and set the status of techcon to "working2"
and return success;
```

Rule six enabled the user to observe this report generation rule in its RITA decompiled form. This form resulted whenever RITA stored statements were displayed to the terminal using the RITA "display" command.

```
rule six
if the status of the techcon is "working2"
then send "^J^J
```

In order to see this rule now in a RITA format, type,

```
*display rule report;    <cr>
```

When the terminal stops typing, then type,

```
*run; <cr>
```

```
in order for the program to continue."  To user
and set status of techcon to "working3"
and return success;
```

Rule seven directed the user to execute the rule, thus generating a trouble ticket report with entries of the user's choosing.

```
rule seven
if the status of the techcon is "working3"
then send "^J^J
```

Now to see how this rule functions. It will ask you, the operating technical controller, certain questions to determine the values that are to be entered into the report format. You may respond with any values - there are no restrictions except that only a single line may be used, and entries should be completed with a carriage return. Running the rule will produce the report, which in this case is a RITA object, named report.

In order for the loaded rule to operate, please type,

```
*create a report whose status is "due"; <cr>
*run;
```

When the terminal stops typing, type the following,

```
*run;    <cr>
```

```
in order for the program to continue."  To user
and set status of techcon to "empty"
and return success;
```

Rule eight explained how to change entries after the report itself was produced.

```
rule eight
if the status of the techcon is "working4"
then send "^J^J
```

If you should desire to modify one of the entries in this report, you could merely repeat the actions above, with the commands,

```
*create a report whose status is "due";    <cr>
*run;    <cr>
```

However, an alternate method would be to type the following,

```
*set the XXXX of the report to  "YYYY";     <cr>
```

where XXXX is the attribute, such as CCSD, find YYYY is the new value. If you desire to try this, also type in,

*display object report; <cr>

in order to verify that the value has indeed been modified. When the terminal stops typing, type in,

*run; <cr>

in order for the program to continue," To user
and set status of techcon to "working 5"
and return success;

Rule nine showed the simplicity of storing the report, a RITA object, in the UNIX data files. It also verified that the report had actually been stored.

rule nine
if the status of the techcon is "working5"
then send "^J^J
Now would be the time to either store or transmit the report, or both. To store the report in an external (to RITA) UNIX file, type,

*display object report to file trubltik; <cr>

where trubltik is the chosen filename.

To verify the file has been stored, you must examine the external UNIX files. The listing of all UNIX files can be viewed by typing:

*shell; <cr>
%ls <cr>

You should see your new filename in the alphabetical listing of all filenames. You can then verify the contents of the file, which should be your trouble ticket report, by typing,

%cat trubltik <cr>

Try the above actions. When the terminal stops typing, then hold down the 'ctrl' key while you press the 'd' key (once only) which should cause the RITA prompt to appear. Then type the following,

```
*run;  <cr>

in order for the program to continue."  To user
and set status of techcon to "working6"
and return success;
```

Rule ten directed the transfer of the report to another computer using the ARPANET File Transfer Protocol (FTP) procedures. Rule ten-a continued the explanation of what comprised the file transfer transaction.

```
rule ten
if the status of the techcon is "working6"
then send "^J^J
    In order to send the report to another computer, an
external file must be called in which contains the various
protocols for establishing the communications links and
transferring the information.  This is all done automatically,
once the following is typed,

*load xmit.report;   <cr>

    When the terminal response is finished and the RITA
prompt is obtained, then type,

*run;   <cr>" to user;
```

```
rule ten-a
if the status of techcon is "working6"
then send "^J^J
    The external file, named xmit.report, establishes an
interconnection with a remote computer and causes a directory
of filenames stored at that computer to be printed out.  It
then transfers a new file (trubltik) to that computer and
again causes the directory to be printed out.  Notice that
the trubltik file has been added.  The communication link
is then terminated.

    When the typing has stopped at the terminal and the RITA
prompt is displayed, then type,

*run;   <cr>

in order for the program to continue.  In the event the
transfer was unsuccessful, you will get a failure message
but the program will continue."  To user
and set status of techcon to "empty"
and return success;
```

Rule eleven showed the ease with which a UNIX data file can be deleted again creating a temporary UNIX shell program.

```
rule eleven
if the status of the techcon is "working7"
then send "^J^J
```

Since reports contain perishable information, sooner or later these stored reports need to be removed from the computer memory. All that is necessary is to type the following,

```
*shell;          <cr>
%del trubltik    <cr>
```

The UNIX system will then ask you to verify whether the file is actually to be deleted. Responding either 'y' for yes or 'n' for no will cause the file to be deleted or retained, respectively. In this case, answer 'y'.

To verify that the file, trubltik, has been deleted, we must again examine the external UNIX files. The listing of all UNIX filenames can be viewed again by typing,

```
%ls          <cr>
```

After performing the above actions, and when the terminal has stopped typing, then hold down the 'ctrl' key while you press the 'd' key (only once) which will cause the RITA prompt to appear. Then type,

```
*run;    <cr>
```

```
in order for the program to continue." To user
and set status of techcon to "working8"
and return success;
```

Rules twelve and thirteen demonstrated how to use the FTP procedure for retrieving a file of a collection of outage reports from a distant computer. In the event the inter-computer connection was unsatisfactory, the rules showed how to obtain the file from a duplicate copy stored in the UNIX data files.

rule twelve
if the status of techcon is "working8"
then send "^J^J
    A special requirement of the ATEC program is to receive data from unique measuring equipment or remote computers. RITA can readily accommodate this need, using retrieve protocols similar to those used earlier for transferring a file.

    To initiate this action, please type,

*load recv.report;   <cr>;

    When the typing has stopped at the terminal and the RITA prompt is displayed, then type,

*run; <cr>

    When the terminal stops typing and the RITA prompt is obtained, please type,

*run; <cr>

In order for the program to continue." To user
and set status of techcon to "empty"
and return success;

rule thirteen
if the status of techcon is "working9"
then send "^J^J
    When the typing has stopped at the terminal and the RITA prompt is shown, then type,

*shell;   <cr>
%cat trubltikfile   <cr>

to examine the contents of this new file. Notice it is a series of trouble ticket reports, in an abbreviated format to save space and slightly modified for RITA to perform the mathematical operations. In the event that the file was not transferred, then you must rename a duplicate copy stored in the UNIX files in order for the program to proceed. Please type,

%cp duplicate trubltikfile   <cr>

    After the UNIX prompt appears, hold down the 'ctrl' key while you press the 'd' key to obtain the RITA prompt. Then type,

*run;   <cr>

in order for the program to continue." To user
and set status of techcon to "working10"
and return success;

Rule fourteen directed the execution of the arithmetic processing of the collection of outage reports to obtain statistical parameters.

```
rule fourteen
if the status of techcon is "working10"
then send "^J^J
    The next portion of this demonstration will illustrate
the use of RITA to perform basic statistical analysis of
data.  We will use the data contained in the file,
trubltikfile, to calculate certain parameters.  Specifically,
the mean and variance of the circuit downtime
will be determined and displayed.  Please type,

*load stats;            <cr>
*load trubltikfile;     <cr>

    When the terminal finishes typing and the RITA prompt
is obtained, type,

*run;     <cr>

    When the typing is finished and the RITA prompt appears,
then type,

*run;     <cr>

in order for the program to continue."  To user
and set status of techcon to "empty"
and return success;
```

Rule fifteen merely advised of the end of agent rules.

```
rule fifteen
if the status of techcon is "working11"
then send "^J^J
    This concludes the interactive demonstration of the
RITA agent for the ATEC automatic report generator.  In
order to terminate this session, please hold down the 'ctrl'
key while you press the 'd' key in order to quit RITA and
return to the UNIX system.  Then type the following to
remove the "trubltikfile" UNIX file:

%rm trubltikfile

You may now logout of UNIX or merely turn off you terminal."
    To user
and set status of techcon to "dead"
and return success;
```

C. ACCESSING THE AGENT

Reference 10 explains the RITA log-in procedures in tutorial fashion in great detail and should be consulted by those users not familiar with RITA or UNIX.

After the log-in was completed, and the UNIX prompt (%) displayed, then the interactive agent was invoked with the command:

```
%rita demo
```

This caused the interactive agent to be loaded and the RITA prompt (*) to appear. Then only the following command was needed to activate the agent:

```
*run;
```

From this point on, the agent operated as described in the previous section.

An actual terminal printout of the agetn execution is included in Appendix A.

VIII. CONCLUSIONS

A. SUMMARY

The RITA agents developed in this thesis, although not operationally complete, readily demonstrated their utility in satisfying the requirements of the ATEC automatic report generator enhancement.

It was observed that RITA was at times quite slow at processing rules and performing calculations. It is assumed that processing time would be improved using RITA agents resident in microprocessors. Even so, RITA seemed to be sufficient for ATEC purposes, especially for small station operations.

B. RECOMMENDATIONS

It is suggested that further investigation be made by ATEC program managers of the potential for utilizing RITA in the ATEC program. The following items require particular attention.

1. Specialized subroutines, especially arithmetic/statistical, should be developed that would enhace the ATEC applications.

2. The procedures for FTP need to be revised to guarantee operational security of data as well as user password and account code.

3. A methodology is needed for handling data groups in other than RITA object format.

APPENDIX A

INTERACTIVE AGENT OUTPUT PRODUCT

This appendix contains an actual terminal printout obtained from executing the interactive agent, "demo," described in chapter VII.

This program demonstrates a capability for using RITA programming techniques to produce an automatic report generator enhancement for the ATEC program. It allows the technical controller to perform the following functions:

A. Define/generate data files automatically according to rules established by the technical controller,

B. Generate/alter rules for building data files,

C. Generate/alter report formats, and

D. Assign rules or mathematical operations on stored data.

Periodically during this demonstration, it will be necessary for you to enter certain commands at the keyboard. You will be notified of what to type. The exact words will be displayed and should be typed in lower case precisely as shown, without the prompt symbol. The carriage return should be typed whenever the symbol, <cr>, is shown.

Should you make a typing error, correct it on the same line using the '#' key to erase each erroneous key stroke, before you type the carriage return. If you discover your error after typing the carriage return, then you must start the demonstration over from the beginning by doing the following:

A. While holding down on the 'ctrl' key, press the 'esc' key. Then release both keys and press the carriage return key. This should cause the terminal to type out the RITA prompt (*).

B. Start the demonstratio over by typing the following on separate lines after receiving either the RITA prompt (*) or UNIX prompt (%), as shown below:

```
*exit;     <cr>
%rita      <cr>
*load demo;     <cr>
*run;      <cr>
```

These same procedures shown above must be followed in the event that either an error or failure message is displayed by the terminal.

First, the demo will show the set-up mode for creating a report format. In this case, a trouble ticket format is prepared that will automatically receive value entries specified by the technical controller. When the RITA prompt (*) appears, please type,

```
*shell;          <cr>
%cat rulereport      <cr>
```

When the terminal stops typing, then hold down the 'ctrl' key while you press the 'd' key (once only) which should cause the RITA prompt to appear. Then type the following,

```
*run;     <cr>
```

in order for the program to continue.
Success!

```
* shell;
% cat rulereport
rule report
if status of report is "due"
then set the name of report to "Trouble Ticket"
and send "What is the ccsd?" to user
and receive next {anything 'ccsd' followed by "^J}  from user
and set ccsd of report to 'ccsd'
and send "What is the source of the report?" to user
and receive next {anything 'source' followed by "^J"} from user
and set source of report to 'source'
and send "What is circuit priority?" to user
and receive next {anything 'priority' followed by "^J"}
    from user
and set priority of report to 'priority'
and send "What is your station?" to user
and receive next {anything 'station' followed by "^J"}
    from user
and set station of report to 'station'
and send "Is the circuit send or receive?" to user
and receive next {anything 'snd/rcv' followed by "^J"}
    from user
and set snd/rcv of report to 'snd/rcv'
and send "Enter the date-time-group of the outage, in the
    form (last digit of calendar year, julian date, zulu
    time)." to user
```

```
and receive next {anything 'dtg-out' followed by "^J"}
    from user
and set dtg-out of report to 'dtg-out'
and send "Enter the date-time-group of festoral, in the form
    (last digit of calendar year, julian date, zulu time)."
    to user
and receive next {anything 'dtg-in' followed by "^J"}
    from user
and set dtg-in of report to 'dtg-in'
and send "What is the code for the reason for outage?"
    to user
and receive next {anything 'rfo-code' followed by "^J"}
    from user
and set rfo-code of report to 'rfo-code'
and send "Enter any desired remarks, limited to one line."
    to user
and receive next {anything 'remarks' followed by "^J"}
    from user
and set remarks of report to 'remarks'
and send concat("^J^J^J^J^J") to user
and display object report
and set status of report to "not-due"
and set status of techcon to "working4"
and return success;
%
* run;
```

The above UNIX file is an example of what the set-up operator would produce to create a new report format. Normal UNIX text editing procedures could then be used to modify the entries of format. Once the format is finally decided upon, then the file is loaded into the RITA system.

Loading a file into the RITA system causes the file to be carefully checked to ensure that the rules and objects have been correctly written. Any errors will be indicated for the user to make corrections. Once all corrections have been made and the file is again loaded into RITA, the system will respond with which rules and objects have been added (to this RITA session in progress) and finish with the RITA prompt.

To see this action, please type,

```
*load rulereport;     <cr>
```

When the terminal stops typing, then type,

```
*run;  <cr>
```

in order for the program to continue.
Success!

```
* load rulereport;
[rule report added]

* run;

    In order to see this rule now in a RITA format, type,

*display rule report;     <cr>

    When the terminal stops typing, then type,

*run; <cr>

in order for the program to continue.
Success!

*display rule report;
rule report:
    IF: the status of the report is "due"
  ThEN: SET the name of the report to "trouble ticket"
        & SEND "What is the ccsd?" to user
        & RECEIVE NEXT {ANYTHING 'ccsd' FOLLOWED BY

"} FROM user
   & SET the ccsd OF the report TO 'ccsd'
   & SEND "What is the source of the report?" TO user
   & RECEIVE NEXT {ANYTHING 'source' FOLLOWED BY
            "

"} FROM user
   & SET the source OF the report TO 'source'
   & SEND "What is circuit priority?" TO user
   & RECEIVE NEXT {ANYTHING 'priority' FOLLOWED BY
            "

"} FROM user
   & SET the priority OF the report to 'priority'
   & SEND "What is your station?" TO user
   & RECEIVE NEXT {ANYTHING 'station' FOLLOWED BY
            "

"} FROM user
   & SET the station OF the report TO 'station'
   & SEND "Is the circuit send or receive?" TO user
   & RECEIVE NEXT {ANYTHING 'snd/rcv/ FOLLOWED BY
            "

"} FROM user
   & SET the snd/rcv OF the report TO 'snd/rcf'
   & SEND "Enter the date-time-group of the outage, in the
        form (last digit)
```

```
OF calendar year, julian date, zulu time)." To user
   & RECEIVE NEXT {ANYTHING 'dtg-out'  FOLLOWED BY
            "

"} FROM user
   & SET the dtg-out OF the report TO 'dtg-out'
   & SEND "Enter the date-time-group of restoral, in the
        form (last digit

OF calendar year, julian date, zulu time)." TO user
   & RECEIVE NEXT {ANYTHING 'dtg-in'  FOLLOWED BY
            "

"} FROM user
   & SET the dtg-in OF the report TO 'dtg-in'
   & SEND "What is the code for the reason for outage?"
        TO user
   & RECEIVE NEXT {ANYTHING 'rfo-code'  FOLLOWED BY
            "

"} FROM user
   & SET the rfo-code OF the report TO 'rfo-code'
   & SEND "Enter any desired remarks, limited to one line."
        TO user
   & RECEIVE NEXT {ANYTHING 'remarks' FOLLOWED BY
            "

"} FROM user
   & SET the remarks OF the report TO 'remarks'
   & SEND concat "

" ) TO user

   & DISPLAY OBJECT report
   & SET the status OF the report TO "not-due"
   & SET the status OF the techcon TO "working4"
   &RETURN SUCCESS;

* run;
```

Now to see how this rule functions. It will ask you, the operating technical controller, certain questions to determine the values that are to be entered into the report format. You may respond with any values - there are no restrictions except that only a single line may be used, and entries should be completed with a carriage return. Running the rule will produce the report, which in this case is a RITA object, named report.

In order for the loaded rule to operate, please type,

```
*create a report whose status is "due";  <cr>
*run;    <cr>
```

When the terminal stops typing, type the following,

```
*run;  <cr>
```

in order for the program to continue.

```
* create a report whose status is "due";
* run;

What is the CCSD?
JJAV9ABC
What is the source of the report?
Maintenance
What is circuit priority?
02
What is your station?
MRE
Is the circuit send or receive?
Send
Enter the date-time-group of the outage, in the form (last
digit of calendar year, julian date, zulu time).
(9,223,1200)
Enter the date-time-group of restoral, in the form (last
digit of calendar year, julian date, zulu time).
(9,224,1345)
What is the code for the reason for outage?
23
Enter any desired remarks, limited to one line.
Circuit altrouted to BBN78GHF, parts are on backorder.

OBJECT report<1>:
        status  IS  "due",
        name    IS  "trouble ticket",
        ccsd    IS  "JJAV9ABC",
        source  IS  "maintenance",
        priority       IS   "02",
        station IS  "MRE",
        snd-rcv IS  "send",
        dtg-out IS  "(9,223,1200)",
        dtg-in  IS  "(9,224,1345)",
        rfo-code       IS   "23",
        remarks IS  "Circuit altrouted to BBN78GHF, parts are
                        on backorder.";

Success!

* run;
```

If you should desire to modify one of the entries in this report, you could merely repeat the actions above, with the commands,

```
*create a report whose status is "due";    <cr>
*run;    <cr>
```

However, an alternate method would be to type the following,

```
*set the XXXX of the report to "YYYY";     <cr>
```

where XXXX is the attribute, such as CCSD, and YYYY is the new value. If you desire to try this, also type in,

```
*display object report;     <cr>
```

in order to verify that the value has indeed been modified. When the terminal stops typing, type in,

```
*run; <cr>
```

in order for the program to continue.
Success!

```
*set the ccsd of the report to "KLJ897TY";

* display object report;
OBJECT report<1>:
        status  IS    "not-due",
        name    IS    "trouble ticket",
        ccsd    IS    "KLJ897TY",
        source  IS    "maintenance",
        priority        IS    "02",
        station IS    "MRE",
        snd/rcv IS    "send",
        dtg-out IS    "(9,223,1200)",
        dtg-in  IS    "(9,224,1345)",
        rfo-code        IS    "23",
        remarks IS    "Circuit Altrouted to BBN78GHF, parts
                          are on backorder.";
```

Now would be the time to either store or transmit the report, or both. To store the report in an external (to RITA) UNIX file, type,

```
*display object report to file trubltik;     <cr>
```

where trubltik is the chosen filename.

To verify the file has been stored, you must examine the external UNIX files. The listing of all UNIX files can be viewed by typing:

```
*shell; <cr>
%ls     <cr>
```

You should see your new filename in the alphabetical listing of all filenames. You can then verify the contents of the file, which should be your trouble ticket report, by typing,

```
%cat trubltik    <cr>
```

Try the above actions. When the terminal stops typing, then hold down the 'ctrl' key while you press the 'd' key (once only) which should cause the RITA prompt to appear. Then type the following,

```
*run; <cr>
```

in order for the program to continue.
Success!

```
* display object report to file trubltik;

* shell;
%ls
demo
duplicate
recv.report
rulereport
stats
trubltik
xmit.report
%cat trubltik
OBJECT report 1 :
        status    IS    "not-due",
        name      IS    "trouble ticket",
        ccsd      IS    "KLJ897TY",
        source    IS    "maintenance",
        priority          IS    "02",
        station   IS    "MRE",
        snd/rcv   IS    "send",
        dtg-out   IS    "(9,223,1200)",
        dtg-in    IS    "(9,224,1345)",
        rfo-code          IS    "23",
        remarks   IS    "Circuit altrouted to BBN78GHF, parts
                          are on backorder.";

%
* run;
```

In order to send the report to another computer, an external file must be called in which contains the various protocols for establishing the communications links and transferring the information. This is all done automatically, once the following is typed,

```
*load xmit.report;    <cr>
```

When the terminal response is finished and the RITA prompt is obtained, then type,

```
*run;     <cr>
```

The external file, named xmit.report, establishes an interconnection with a remote computer and causes a directory of filenames stored at that computer to be printed out. It then transfers a new file (trubltik) to that computer and again causes the directory to be printed out. Notice that the trubltik file has been added. The communication link is then terminated.

When the typing has stopped at the terminal and the RITA prompt is displayed, then type,

```
*run;     <cr>
```

in order for the program to continue. In the event the transfer was unsuccessful, you will get a failure message but the program will continue.
Success!

```
* lord xmit.report;
[Object send.report<1> added]
[Object portl<1> added]
[Rule ftp added]
[Rule sendhostname added]
[Rule senduser added]
[Rule sendpass added]
[Rule sendpassword added]
[Rule sendacct added]
[Rule sendacctcode added]
[Rule directory1 added]
[Rule sendfile added]
[Rule directory2 added]
[Rule logout added]

* run;
 "

 Unknown host name
Host: Connections established" "

230 login completed

"OBJECT portl<1>:
        response          IS    ".
> 151 <NPS-ACCAT>
151 $XED-MODE-FILE$.;1
151 TRUBLTIKFILE.;3
151 ^v^+]ARCHIVE";
```

```
-DIRECTORY[.;1
200 End of status.
> > 200 Allocation ignored on this system.
255 SOCK 3276899858
250 Store of <NPS-ACCAT>TRUBLTIK.;1;P775200;ACC-, ASCII type,
    started.
252 Transfer completed

"OBJECT portl<l>:
        response        IS    ".
> > 151 <NPS-ACCAT>
151 $XED-MODE-FILE$.;1
151 TRUBLTIK.;1
151 TRUBLTIKFILE.;3
151 ^v^+]ARCHIVE";

"

-DIRECTORY[.;1
200 End of staus.
> > % % 231 BYE command received

"Success!

* run;
```

Since reports contain perishable information, sooner or later these stored reports need to be removed from the computer memory. All that is necessary is to type the following,

```
*shell;          <cr>
%del trubltik    <cr>
```

The UNIX system will then ask you to verify whether the file is actually to be deleted. Responding either 'y' for yes or 'n' for no will cause the file to be deleted or retained, respectively. In this case, answer 'y'.

To verify that the file, trubltik, has been deleted, we must again examine the external UNIX files. The listing of all UNIX filenames can be viewed again by typing,

```
%ls              <cr>
```

After performing the above actions, and when the terminal has stopped typing, then hold down the 'ctrl' key while you press the 'd' key (only once) which will cause the RITA prompt to appear. Then type,

```
*run;      <cr>
```

in order for the program to continue.
Success!

```
* shell;
% del trubltik
trubltik
--delete? y
% ls
demo
duplicate
recv.report
rulereport
state
xmit.report
%
* run;
```

A special requirement of the ATEC program is to receive data from unique measuring equipment or remote computers. RITA can readily accommodate this need, using retrieve protocols similar to those used earlier for transferring a file.

To initiate this action, please type,

```
*load recv.report;     <cr>;
```

When the typing has stopped at the terminal and the RITA prompt is displayed, then type,

```
*run;    <cr>
```

When the terminal stops typing and the RITA prompt is obtained, please type,

```
*run;    <cr>
```

in order for the program to continue.
Success!

```
* load recv.report;
[Object send.reports<1>added]
[Object port]<1> added]
[Rule ftps added]
[Rule sendhostnames added]
[Rule sendusers added]
[Rule sendpasss added]
[Rule sendpasswords added]
[Rule sendaccts added]
[Rule sendacctcodes added]
[Rule recvfile added]
[Rule logouts added]

*  run;
 "
```

```
 Unknown host name
Host: Connections established" "

230 login completed

" "


.
> 255 SOCK 3276899859
250 ASCII retrieve of <NPS-ACCAT>TRUBLTIKFILE.;3 started.
252 transfer completed" "

.
> > 231 BYE command received

"Success!

* run;
```

When the typing has stopped at the terminal and the RITA prompt is shown, then type,

```
*shell;     <cr>
%cat trubltikfile     <cr>
```

to examine the contents of this new file. Notice it is a series of trouble ticket reports, in an abbreviated format to save space and slightly modified for RITA to perform the mathematical operations. In the event that the file was not transferred, then you must rename a duplicate copy stored in the UNIX files in order for the program to proceed. Please type,

```
%cp duplicate trubltikfile     <cr>
```

After the UNIX prompt appears, hold down the 'ctrl' key while you press the 'd' key to obtain the RITA prompt. Then type,

```
*run;     <cr>
```

in order for the program to continue.

```
Success!

* shell;
% cat trubltikfile

object report
status is "due",
ccsd is "JJAV9ABC",
dtg-in is ("9","362","22","15"),
dtg-out is ("9","361","09","00");
```

```
object report
status is "due",
ccsd is "CGSD89JK",
dtg-in is ("9","356","23","15"),
dtg-out is ("9","355","12","55");

object report
status is "due",
ccsd is "JJAV9ABC",
dtg-in is ("9","345","15","00"),
dtg-out is ("9","344","10","10");

object report
status is "due",
ccsd is "CVFG45JK",
dtg-in is ("(","333","18","50"),
dtg-out is ("9","332","15","35");

object report
status is "due",
ccsd is "JJAV9ABC",
dtg-in is ("9","322","03","00"),
dtg-out is ("9","322","01","10"");

object report
staus is "due",
ccsd is "JJAV9ABC",
dtg-in is ("9","289","20","00"),
dtg-out is ("9","279","21","05");

object report
status is "due",
ccsd is "JJAV9ABC",
dtg-in is ("9","267","15","15"),
dtg-out is ("9","266","04","40");

object report
status is "due",
ccsd is "JJAV9ABC",
dtg-in is ("9","237","23","40"),
dtg-out is ("9","235","02","35");

object report
status is "due",
ccsd is "JJAV9ABC",
dtg-in is ("9","195","01","20"),
dtg-out is ("9","194","20","55");

object report
status is "due",
ccsd is "JJAV9ABC",
dtg-in is ("9","001","13","40"),
dtg-out is ("8","364","18","45");
%
* run;
```

The next portion of this demonstration will illustrate the use of RITA to perform basic statistical analysis of data. We will use the data contained in the file, trubltikfile, to calculate certain parameters. Specifically, the mean and variance of the circuit downtime will be determined and displayed. Please type,

```
*load stats;            <cr>
*load trubltikfile;     <cr>
```

When the terminal finishes typing and the RITA prompt is obtained, type,

```
*run;    <cr>
```

When the typing is finished and the RITA prompt appears, then type,

```
*run;     <cr>
```

in order for the program to continue.
Success!

```
* load stats;
[Object statistic<1> added]
[Object downtime<1> added]
[Rule stats1 added]
[Rule stats2 added]
[Rule stats3 added]
[Rule stats4 added]
[Rule stats5 added]
[Rule stats6 added]
[Rule stats7 added]
[Rule stats8 added]
[Rule stats9 added]
[Rule stats10 added]
[Rule stats11 added]
[Rule stats12 added]
[Rule stats13 added]
[Rule stats14 added]
[Rule stats15 added]

* load trubltikfile;
[Object report<2> added]
[Object report<3> added]
[Object report<4> added]
[Object report<5> added]
[Object report<6> added]
[Object report<7> added]
[Object report<8> added]
[Object report<9> added]
[Object report<10>added]
[Object report<11>added]
```

```
* run;
OBJECT downtime<1>:
        ccsd    IS    "JJAV9ABC",
        variance          IS   "18419354.686 minutes squared",
        mean    IS    "2 days, 9 hours, 13.75 minutes";

Success!

* run;
```

This concludes the interactive demonstration of the RITA agent for the ATEC automatic report generator. In order to terminate this session, please hold down the 'ctrl' key while you press the 'd' key in order to quit RITA and return to the UNIX system. Then type the following to remove the "trubltikfile" UNIX file:

```
%rm trubltikfile  <cr>

You may now logout of UNIX or merely turn off your terminal.
Success!
```

APPENDIX B

LIST OF KEY ABBREVIATIONS

ARPA	-	Advanced Research Projects Agency
ATEC	-	Automated Technical Control
DCA	-	Defense Communications Agency
DCS	-	Defense Communications System
FACC	-	Ford Aerospace and Communications Corporation
LRIP	-	Low Rate Initial Production
RITA	-	Rule-directed Interactive Transaction Agent
SYSCON	-	System Control
TCF	-	Technical Control Facility

BIBLIOGRAPHY

1. Adams, F. A., Rand Intelligent Terminal Agent (RITA) Applications in Telecommunications Management, Master's Thesis, Naval Postgraduate School, March, 1979.

2. Air Force Communications Service Report, Automated Technical Control (ATEC) Benefits Analysis, AFCS AdHoc Action Team, June 1978.

3. Defense Communications Agency Report, Concept of Operations for Automated Technical Control (ATEC), Hq AFCS/DOY, 14 June 1978.

4. Department of the Air Force Report, ATEC Base Line Requirements Document (BLRD), Hq AFCS, 1 October 1976.

5. Kernighan, B. W., A Tutorial Introduction to the UNIX Text Editor, internal memorandum at Bell Laboratories, Murray Hill, New Jersey, undated.

6. Kernighan, B. W., UNIX for Beginners, paper presented at Bell Laboratories, Murray Hill, New Jersey, 10 March 1976.

7. Mitre Corporation Report, Automated Technical Control (ATEC) Data Processing System Description, 11 September 1978.

8. Rand Report R-1808-ARPA, RITA Reference Manual, by R. H. Anderson and others, September 1977.

9. Rand Report R-1908-ARPA, RAND Intelligent Terminal Agent (RITA): Design Philosophy, by R. H. Anderson and J. J. Gillogly, February 1976.

10. Rand Report, RITA Newsletter Number 2, by William Giarla, December 1978.

11. Warren, T. E., A User's Guide for the RITA Production Rule System, Master's Thesis, Naval Postgraduate School, March, 1978.

12. Waterman, D. A., An Introduction to Production Rule Systems, paper presented at the Rand Corporation, Santa Monica, California, November 1976.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93940	2
3.	Professor John M. Wozencraft, Code 74 Chairman, C^3 Academic Group Naval Postgraduate School Monterey, California 93940	3
4.	Professor G. K. Poock, Code 55Pk Department of Operations Research Naval Postgraduate School Monterey, California 93920	25
5.	Maj Mark H. Smith, USAF 9500 Braddock Road Fairfax, Virginia 22032	1
6.	Hq AFCS/OA Attn: Mr. Charles Jacobs Scott AFB, Illinois 62225	2
7.	Hq AFCS/DOYT Attn: Maj Carl Huebner Scott AFB, Illinois 62225	1
8.	Lt Col R. J. Roland, USAF, Code 52R1 Department of Computer Science Naval Postgraduate School Monterey, California 93940	1
9.	Lt Col F. R. Murphy, USAF, Code 39 C^3 Curricular Officer Naval Postgraduate School Monterey, California 93940	1
10.	AFIT/CIRD Attn: Capt Cain Wright-Patterson AFB, Ohio 45433	1

11. Department Chairman, Code 55 — 1
Department of Operations Research
Naval Postgraduate School
Monterey, California 93940

12. Professor D. R. Barr, Code 55Bn — 1
Department of Operations Research
Naval Postgraduate School
Monterey, California 93940

13. Mr. C. Blais — 1
Naval Ocean Systems Center
Code 8321
San Diego, California 92152

14. Mr. R. Brandenburg — 1
Naval Ocean Systems Center
Code 8321
San Diego, California 92152

15. Dr. M. Denicoff, Code 437 — 1
Office of Naval Research
800 N. Quincy Street
Arlington, Virginia 22217

16. LCDR A. J. Dietzler — 1
Advanced Research Projects Agency
Information Processing Techniques Office
1400 Wilson Boulevard
Arlington, Virginia 22209

17. Professor D. P. Gaver, Code 55Gv — 1
Department of Operations Research
Naval Postgraduate School
Monterey, California 93940

18. Mr. H. Miller — 1
Naval Electronics Systems Command
PME-108-3
Washington, D. C. 20360

19. Mr. D. Norman, Code 0141 — 1
Manager of Operations
W. R. Church Computer Center
Naval Postgraduate School
Monterey, California 93940

20. Assoc Professor S. H. Parry, Code 55Py — 1
Department of Operations Research
Naval Postgraduate School
Monterey, California 93940

21. Professor G. A. Rahe, Code 52Ra — 1
Department of Computer Science
Naval Postgraduate School
Monterey, California 93940

22. Assoc Professor F. R. Richards, Code 55Rh — 1
Department of Operations Research
Naval Postgraduate School
Monterey, California 93940

23. COL D. Russell, USA — 1
Advanced Research Projects Agency
Information Processing Techniques Office
1400 Wilson Boulevard
Arlington, Virginia 22209

24. Dr. J. Schill — 1
Naval Ocean System Center
Code 8123
San Diego, California 92152

25. Professor N. F. Schneidewind, Code 54Ss — 1
Department of Administrative Sciences
Naval Postgraduate School
Monterey, California 93940

26. Mr. C. C. Stout — 1
Naval Electronics Systems Command
Code 330
2511 Jefferson Davis Highway
Arlington, Virginia 20360

27. Dr. M. Tolcott, Code 455 — 1
Office of Naval Research
800 N. Quincy Street
Arlington, Virginia 20360

28. CAPT R. D. Yingling, USAF, Code 62Ya — 1
Department of Electrical Engineering
Naval Postgraduate School
Monterey, California 93940

29. Mr. Bill Carper — 1
Naval Ocean Systems Center
Code 832
San Diego, California 92152

30. CDR. Don Lesemann — 1
Naval Ocean Systems Center
Code 832
San Diego, California 92152

31. Mr. Garry Martins 1
Rand Corporation
1700 Main St.
Santa Monica, California 90406

32. Mr. William Giarla 1
Rand Corporation
1700 Main St.
Santa Monica, California 90406